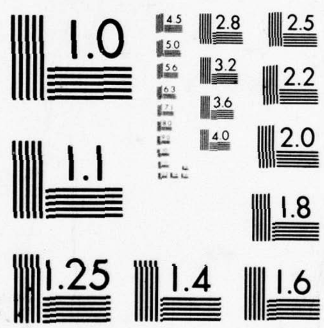
AD-A053 197 WHARTON SCHOOL PHILADELPHIA PA DEPT OF DECISION SCIENCES F/G 9/2
COST EFFECTIVE DATABASE DESIGN: AN INTEGRATED MODEL.(U)
JAN 77 R GERRITSEN N00014-75-C-0462
UNCLASSIFIED 77-01-09 NL

1 OF 1
AD
A053197

END
DATE
FILMED
6-78
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(11)

COST EFFECTIVE DATABASE DESIGN:
AN INTEGRATED MODEL

Rob Gerritsen

77-01-09

Department of Decision Sciences
The Wharton School
University of Pennsylvania
Philadelphia, Pennsylvania 19174

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 14 77-01-09	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 Cost Effective Database Design: An Integrated Model,		5. TYPE OF REPORT & PERIOD COVERED 9 Final rept.,
		6. PERFORMING ORG. REPORT NUMBER 77-01-09
7. AUTHOR(s) 10 Rob Gerritsen		8. CONTRACT OR GRANT NUMBER(s) 15 N00014-75-C-0462
9. PERFORMING ORGANIZATION NAME AND ADDRESS Decision Sciences Department University of Pennsylvania/Wharton School Philadelphia, PA 19104		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 9 Technical report.,
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Information Systems Arlington, VA 22217		12. REPORT DATE 1/77
		13. NUMBER OF PAGES 15 11 Jan 77
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 12 14p.
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Unlimited

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Database
Plex structures
Network structures

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Relationships between conceptual schema design and internal schema design are considered. The author's earlier work on automatic derivation of conceptual schemata is reviewed, and a constrained optimization model ~~model~~ for design of internal schemata is introduced. Trade-offs represented in the model are discussed, and a method for obtaining access frequency estimates is described.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

COST EFFECTIVE DATABASE DESIGN:
AN INTEGRATED MODEL

Rob Gerritsen
Department of Decision Sciences
The Wharton School
University of Pennsylvania
Philadelphia, Pennsylvania 19174

ABSTRACT

Relationships between conceptual schema design and internal schema design are considered. The author's earlier work on automatic derivation of conceptual schemata is reviewed, and a constrained optimization model for design of internal schemata is introduced. Trade-offs represented in the model are discussed, and a method for obtaining access frequency estimates is described.

1.0 INTRODUCTION

The data base design problem is a formidable one. There are many inputs that the Data Base Administrator (DBA) must consider. In addition to the logical requirements that the data base must satisfy, the DBA must consider the characteristics of the media on which the data base is to be stored, the costs over time of using the various media, the costs of using computer time to process the data base, the physical limitation of the storage media, the characteristics of the data such as the frequencies of various types of data occurrences, and the usage characteristics of the data base.

The ANSI/X3/SPARC study group has defined three types of schema: conceptual, internal and external [1]. We see the conceptual schema as being derived from a set of required outputs (henceforth loosely termed the set of queries) that are accumulated from the entire user community prior to beginning data base design. That is, each user must specify a set of queries that describe his informational needs. Then all of these sets of queries are combined for the algorithm that designs the data base. This algorithm has been previously described by us [5] and other than presenting a brief example, we will not go into it here.

After the conceptual schema has been completed, it then becomes possible to bring into consideration various physical characteristics of the data base. These characteristics, mentioned above, include media characteristics, volumes of data, and frequencies of access. Application of these constraints and costs to the conceptual schema makes it possible to derive an internal schema.

In this paper we develop a constrained optimization model that derives an internal schema, given that the DBMS is of the CODASYL DBTG type [2]. We proceed by first presenting a prose description of the decisions, knowledge and tradeoffs that must be represented in the model. Then the model itself is presented.

2.0 MATHEMATICAL MODELLING FOR DATABASE DESIGN

The application of mathematical models to database and, earlier, file design, is certainly not a new idea. However, most such models for database design suffer from a lack of generality and integration. In addition, some models that attempt to be general end up being too general, and one attempting to apply such models is faced with the difficult task of mapping the database concepts of the system he uses to

the general concepts used in the model. An example of too much generality is the model presented by Yao and Merten [14]. This model recognizes only two access methods, a dichotomy also proposed by Severance [12], physically sequential and logically sequential (i.e. using pointers). Although many file organizations can be decomposed to these basic access methods, most DBA's think in terms of higher level concepts and structures that combine these access methods, such as hashing, chain, pointer array, ISAM, etc. The Yao and Merten model is also constrained to one file at a time, and that file can contain only one record type. This, as the authors state, limits the model to file design rather than database design.

Mitoma and Irani [10] developed a design optimization model specifically for a system built according to the CODASYL Data Base Task Group (DBTG) [2] specifications, as we will do here. However, the Mitoma and Irani model incorporates as variables only the implementation choices for the relationships (sets). Choices relating to location modes are not incorporated in the model. The objective function contains only access costs (storage costs appear in a constraint), and a user of the model is on his own with respect to the determination of various coefficients such as the access costs associated with each alternative implementation for each relation. Since such access costs are dependent on the other design choices that are to be made, a model should not assume them to be given.

More recently, De, Haseman and Kriebel [4] have developed a model that is very similar to the one due to Mitoma and Irani. This model is more sophisticated with regard to costs, clarifying how they are to be derived. However, like its predecessor, the only decision variable incorporated in the model is the choice of set implementation. The model has an interesting additional constraint associated with an upper bound on the expected access time to process a query.

Martin [9] presents detailed analysis of trade-offs in various file organizations, search techniques, addressing schemes, etc. Others [11, 8] have focused on index or file inversion techniques. These models serve well if one wishes to select only one addressing scheme, one indexing technique, one search technique, one etc. However, since a DBMS may uses several techniques from each category in combination, an integrated model that recognizes cross-technique effects is needed.

3.0 CONCEPTUAL SCHEMA DESIGN

For purposes of illustration we have included in Figure 1 a conceptual schema that was derived from a set of four queries also illustrated in the figure. One can see that the conceptual schema reflects the relationships that exist between the different items that are stored in the data base. These relationships are of three types. A one-to-one relationship results in placing the items that participate in the relationship into a single record type. For example, patient-number and patient-name are in a one-to-one relationship. A one-to-many relationship results in the creation of a hierarchy (SET in DBTG terminology) in the data base. For example, patient-number is in a one-to-many relationship with bill-amount.

"For a given Doctor-name list all values of patient-name, patient-number, hospital-name, and diagnosis,"

"For a given patient-number list the value of patient-name, hospital-name and all values of diagnosis and doctor-name."

"For a given paitient-number list all values of bill-amount and bill-code."

"For all bills of a given code, list the values of bill-amount and patient-number."

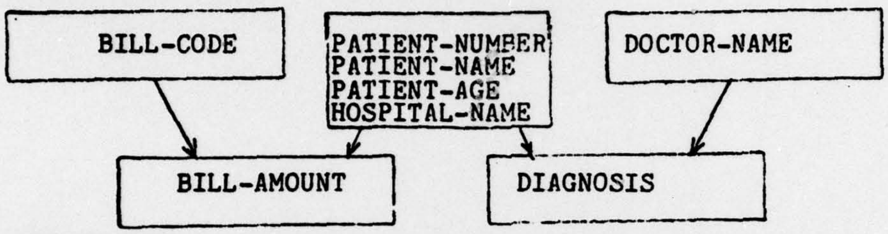


Figure 1. Example of a conceptual schema from a set of queries.

The third kind of relationship is a many-to-many relationship. This relationship is really two one-to-many relationships. For example, there is a manymto-many relationship between patient-name and doctor-name. That is, one can associate many doctor-names with a single patient-name, and simultaneously, one can also associate many patient-names with a single doctor-name. The design algorithm can also recognize the need for file inversions (or indexes) by the presence in one or more queries of an equivalence condition on a particular data item.

When multiple simultaneous relationships exist it becomes necessary to label each relationship, a feature not illustrated in Figure 1.

In summary, the conceptual schema design phase determines the contents of the record types, the relationships between the record types, and possible file inversions.

4.0 INTERNAL SCHEMA DESIGN

4.1 DECISIONS TO BE MADE

The internal schema can be designed once the conceptual schema has been completed. This will require several design choices. One such choice is to determine how each record type's location in the data base is to be determined when an occurrence is first stored. In DBTG terminology this is known as the LOCATION MODE. In addition, an implementation choice has to be made for each relationship that is to be represented in the data base. In DBTG terminology this is know as SET MODE. There are two standard set modes defined in the DBTG report. One is known as CHAIN and the other is known as POINTER ARRAY. If chain mode is selected, the DBA must also determine whether certain optional linkages are to be included.

Several decisions have to be made regarding the media on which the data base is to be stored. One such decision involves the subdivision of the database into areas. An area is a physical chunk of the data base, typically a file at the operating system level. For each area the DBA must determine what is to be the size of the pages of that area. A page is the equivalent of what is sometimes called a bucket. The size of the page also determines how much data is transferred whenever an access to the data base is made. ence, page size usually is an integral multiple of the block size of the media on which the area is stored. Other

decisions are the number of pages for each area, and which media to use for each area.

Finally, there are two "yes-no" decisions that the DBA must make. For each record type in the data base he may optionally request a singular set definition. In DBTG terminology a singular set is known as a SYSTEM owned set. Through a singular set it is possible to access all occurrences of a particular record type. Hence, one would use it if such queries were common and if the number of occurrences of that record type were not too large. The second "yes-no" decision is what we call a flattening decision for each relationship in the data base. By flattening is meant, simply, the removal of all explicit structure (i.e., pointers) and instead capturing the hierchical relationship through replication of data values from the owning record in each member record of the hierchical relationship. The process of flattening is similar to the <u>reverse</u> of normalization in the relational model [3]. Figure 2 illustrates this concept. The flattening decision has not yet been incorporated in the mathematical model that follows, but we have discussed some of the modelling considerations elsewhere [6].

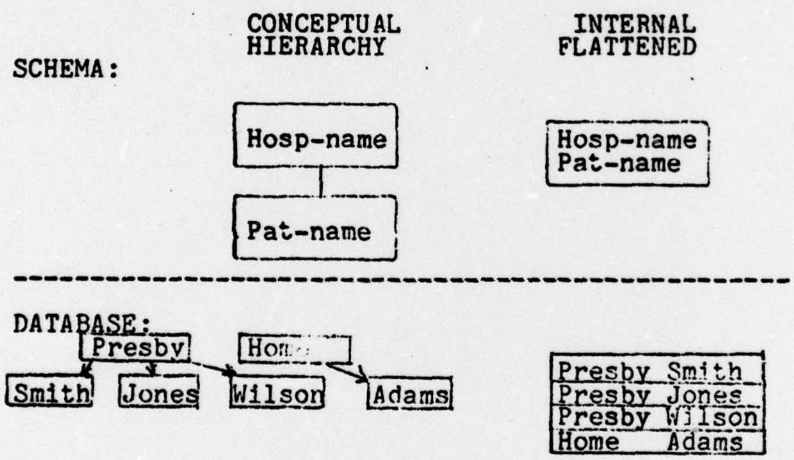


Figure 2. Flattening of hierarchies.

At this point it is appropriate to detail some of the assumptions which will hold for the rest of this paper. The first (should be obvious by now) is that we will operate in a DBTG environment. In addition, whenever it is necessary to assume a particular style of implementation of the DBTG specifications, we have assumed an implementation that is similar to the WAND [7] implementation. Another assumption is that variable length records are not permitted. We also assume that only one member type record is permitted in a set definition. A final important assumption is that our model utilizes only expected values for stochastic variables. Since interactions of the variables are not linear, this assumption will require further analysis.

We also ignore several decisions that one could make that might have an impact on data base efficiency. Such decisions include whether or not to use duplicate CALC keys in the data base, whether or not to use Boolean operations on keys, and whether or not to use sorted sets. An important consideration in data base design is the potential growth of the data base. This is another factor we ignore.

4.2 KNOWLEDGE TO BE USED

What is the knowledge that should be used when designing an internal schema? First, of course, we must use the conceptual schema derived in the first phase of design. Secondly, costs are important, including access costs, storage costs, CPU costs, and access times. Block sizes of the various media must be used, the data volumes (that is the number of occurences of each record type) must be used, and the data sizes (that is the size in bytes of each record type) must be used. Since structure is implemented with pointers, overheads due to pointers must be considered. Storage capacity of each medium becomes a constraint. Finally, one of the most important knowledge inputs is access frequencies.

We propose to derive the access frequencies during the input phase of the conceptual schema design process. That is, when the users are specifying the queries that they want answered from the data base. Let's illustrate how this is to be done. In Figure 3 is a graphical representation of a query. The nodes marked I=1, I=2, etc., are representative of record types and the arcs marked J=6, J=7, etc., are representative of set types. Hence, figure 3 illustrates a query that begins by accessing a record of type 1 and then accesses a record of type 2 through the set of type 6 followed by an access to the record of type 3 through the set of type 7, etc. Note that we have drawn a line through the arc marked J=7 and another line through the arcs marked J=8 and J=9. These arcs represent conditional tests that are present in the query. For example, retrieval of the record of type 2 permits evaluation of a predicate based on the data values in either records of type 1 and/or records of type 2. Whenever the user specifies a condition in his query, we will ask him what is his expectation for the success of that test. For the first test in our example the user has specified that he expects a 60% rate of success. There is a second test in this query following the access to record of type 3 and the user has indicated a 20% expected rate of success.

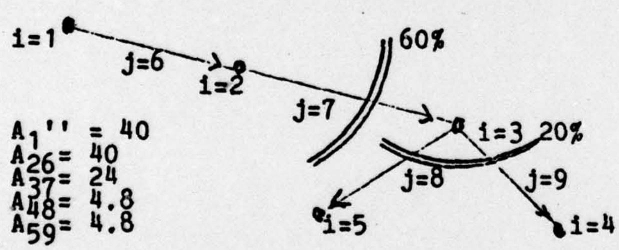


Figure 3. Access path tree and derived access frequencies.

One problem with other database design models that do not use the set of expected queries as one of the inputs is that those model must use estimation at a very gross level to determine access frequencies. Such estimation is compounded by joint and disjoint probabilities. Tversky and Kahnemann [13] have demonstrated that people tend to dramatically overestimate joint and underestimate disjoint probabilities. Although these errors are compensating when overall access frequencies are being estimated, we still would not have a high level of confidence in such estimates. The method used here asks only that people estimate single, not joint and not conditional, probabilities.

For each query the user must also predict how frequently over some time period t (say a month) the query is to be executed. If the user replies for example, forty times, then we can conclude that due to the query in Figure 3, a record of type 1 will be accessed forty times per month. In addition, using the single probabilities provided by the user, we can conclude that there will be 40 accesses to records of type 2 through set type 6, that there will be 24 accesses to record type 3 through set type 7, and 4.8 accesses to record types 5 and 4 through set types 8 and 9

respectively.

We define three terms to describe three different types of accesses that can occur in the data base. A_{ij} describes the number of accesses during period t to record i through set j. A_i' describes the number of accesses during time period t to record i with no other information. We call such an access a "cold" access. A_i'' describes the number of accesses during time period t to record i using a CALC key. Then for figure 3, assuming the initial access is a CALC access, $A_1''=40$, $A_{26}=40$, $A_{37}=24$, $A_{48}=4.8$, and $A_{59}=4.8$.

Our illustration considered only one query. As was stated before, the data base is designed from a large set of queries from an entire group of users. Therefore these A_{ij}, A_i', and A_i'' are accumulated over all values determined from all of the queries.

Several comments are appropriate. Note that the A_{ij} represent access through set _types_. If record type _i_ is a member of set type _j_, then we assume that the access will be to _all_ member occurrences of a set occurrence of set type _j_. On the other hand, if _i_ is the owner of _j_, then we assume that only a single record occurrence will be accessed. Derivation of record occurrence accesses from record type accesses is accomplished in the model (and described below) through application of the conceptual schema and relative record frequencies.

The time period t should be sufficiently large to be representative of all types of processing. A value of one month or one year would be appropriate. Another approach is to let the user select the appropriate time period for each query leaving the job of normalization to a single period up to the system.

It might also be appropriate for the system to interact with the user for consistency checking. It is not at all unlikely that similar predicates will appear in different queries, perhaps from different users. In such cases, if the estimates of success vary, the system could point this out and ask for resolution. Similarly, if the system knows ranges of possible values for items then certain probabilities might appear suspect. For example a 1% estimated probability of success for a predicate involving a three-valued item could be questioned. Finally, the system can estimate the size of each report that results from a query and ask the user if such a size appears to be reasonable.

4.3 TRADE-OFF ANALYSIS

Let us turn now to a description of the various effects and trade-offs that should be considered for inernal schema design. First, we will consider the various methods of implementation of relationships. As we mentioned before, there are two principle methods known as CHAIN and POINTER ARRAY. The chain mode has some additional options.

Chains always use NEXT pointers, but may also optionally contain PRIOR pointers and OWNER pointers. A chain with only next pointers is a simple list with the last element of the list pointing back to the first element on the list. A chain with next and prior pointers is a doubly linked list. And a chain with next, prior and owner pointers also has a pointer to the beginning of the list from every element on the list except the first (the first being the owner).

The basic trade-offs here are storage and time. A chain with next, prior, and owner pointers requires approximately three times as much pointer space, but may be much more efficient in terms of retrieval time when we are trying to process the records in that relationship. Prior pointers are especially valuable for efficiency when updating. One can easily verify that it is much easier to insert or remove elements from the middle of a doubly linked list rather than a simple list.

A second major consideration is selection of a record location mode for each record type that exists in the data base. The location mode can have a dramatic effect on the access speed when a record is retrieved. There are two location modes.* One is the CALC location mode which to many is known as the hashed location mode. The other location mode is known as the VIA location mode. A record type that has a VIA location mode must be VIA a particular relationship (set) of which the record type is a member. When a record occurrence that has a VIA location mode is first placed into storage it will be placed on the same page as that page that contains the record occurrence that owns the VIA set. If there is not room on that page it is placed physically close to that page. Figure 4 presents a table illustrating the two choices of access mode and their interaction with four different accessing styles that might occur during the use of the data base. These four styles are one greater than the three previously introduced. This is so because there is a big diffrence in accesses through a set if the set _is_ the VIA set.

	ACCESS TYPE			
	CALC	Thru VIA Set	Not Thru VIA Set	Cold
LOCATION MODE:				
CALC	1	X	3	4
VIA Set	X	2	3	4

Figure 4. Interaction of access type with LOCATION MODE. (See text for explanation of code numbers.)

Let's now describe what happens (type numbers correspond to the numbers in Figure 4.) Type 1 interaction: If the record's location mode is CALC, and a CALC accesss is attempted then it may be possible to find that record in only one access to the CALCulated page. However, it is well known that overflows occur even when the data base is only partially loaded, and with overflows additional accesses are required.

Type 2 interaction: If location mode is VIA and access is through this set then one may be able to find all of the members of the set without any physical accesses occurring. This happens because in order to logically access the members one had to first physically access the owner. Since the members may all be stored on the same page with the owner, the access to the owner has also already physically accessed the members. Further physical accesses in this case are only required if the members of the set are so numerous that they are not all on the same page with the record that owns the set, or if other records on the same page force overflow of the members of the VIA set.

Type 3 interaction: If one is accessing a record through some set which is not the VIA set then it doesn't matter whether the location mode is CALC or VIA, since one would expect to have to access a different page for every member record occurence. Note: This is not actually true because occurrences of this record type may be clustered in the data base. Even though one is not utilizing the clusters explicitly, it is possible that these clusters will coincidentally reduce the physical to logical access ratio. Such _secondary clustering effects_ could be estimated, but we have not yet done so in our model.

Type 4 interaction: In this case location mode also makes little difference. To access records that are not CALC and without going through a set requires either that one does an exhaustive search of the area(s) that contains all occurrences of the record types, or that one accesses the record through a system owned (singular) set. Note that if such other accesses are frequent for a particular record type we might

- - - - - - - -

* DBTG recognizes a third, called DIRECT, which we have ruled out because if appears to be a poor design feature, making data bases un-restructurable.

choose to segregate that record type to a particular area. This would result in a relatively small area, perhaps making it cost effective to search that area.

Let's examine some other trade-offs associated with area size, page size and the core buffer. Larger areas will obviously increase storage cost. A larger area will also mean higher search costs if one ever needs to exhaustively search that area. On the other hand, a smaller area will tend to increase the probability of overflows. Larger pages will require more data transfer on every I/O operation. Larger pages also require larger core buffers. On the other hand, smaller page sizes mean more overflows and an increase in unused space. A larger core buffer, containing more pages in fast direct memory, means that one must use more core memory and incur an expense. On the other hand, a smaller core buffer decreases the probability that a record that is desired is already in fast core and hence will result in more accesses.

4.4 A CONSTRAINED OPTIMIZATION MODEL

4.4.1 Index Sets -

$R = \{1,\ldots,i,\ldots n\}$ RECORD INDEX SET
$S = \{1,\ldots,j,\ldots m\}$ 'SET' INDEX SET
$L = \{1,\ldots,j,\ldots m,\ m+1\}$ LOCATION MODES
$U = \{1,\ldots,k,\ldots u\}$ 'SET' IMPLEMENTATION

The index sets R, S, and L are derived from the conceptual schema: $\underline{n}$ record types and $\underline{m}$ set types (relationships). There are $\underline{m+1}$ location modes; $\underline{m}$ of these are VIA set $\underline{j}$ and one is the CALC location mode. The index set U is dependent on the implementation of the particular DBMS that is to be used. In the WAND system u=5; sets can be implemented in one of five ways:

1. Chain with NEXT pointers only.
2. Chain with NEXT and OWNER pointers.
3. Chain with NEXT and PRIOR pointers.
4. Chain with NEXT, PRIOR and OWNER pointers.
5. Pointer Array.

4.4.2 Decision Variables -

x_{ij} , $i \in R$, $j \in L$. If $x_{ij} = 1$, then the i^{th} record has j^{th} location mode.
y_{jk} , $j \in S$, $k \in U$. If $y_{jk} = 1$, then the j^{th} set has k^{th} implementation.
z_i , $i \in R$. If $z_i = 1$, then the i^{th} record is in a singular set.
p Number of pages in database.
b Number of pages in core (Buffer).

The decision variables presented here do not include subdivision of the database into areas, and hence assignment of record types to areas is also not considered. We are presently extending the model to incorporate these decisions as well.

4.4.3 Given -

O_{ij} , $i \in R$, $j \in S$. If $O_{ij}=1$, then the i^{th} record owns the j^{th} set.

M_{ij} , $i \in R$, $j \in S$. If $M_{ij}=1$, then the i^{th} record is a

member of the j^{th} set.

B is the block (page) size in bytes.

G_1 is the capacity of secondary storage in bytes.

G_0 is the capacity of primary storage in bytes.

F_i, $i \varepsilon R$, is the number of occurrences of the i^{th} record type.

D_i, $i \varepsilon R$, is the data size in bytes of the i^{th} record type.

C_1 is the cost to store 1 block for time period t on secondary storage.

C_0 is the cost of one byte-second of primary storage.

T is the average time, in seconds, for one access to secondary storage.

A_{ij} is the number of accesses to record i through set j during time period t.

A_i' is the number of accesses to record i requiring search during time period t.

A_i'' is the number of accesses to record i using a calculated key during time period t.

$$E_j = \frac{\sum_{i \varepsilon R} M_{ij} F_i}{\sum_{i \varepsilon R} O_{ij} F_i}$$

E_j is the expected number of members for each occurrence of set j.

Q_k, $k \varepsilon U$; $Q_k=1$ implies that set type k has owner pointers. In WAND $Q=\{0,1,0,1,1\}$.

KM_k, $k \varepsilon U$, is the number of bytes in a member record occurrence to implement set type k. For WAND $K_m=\{5,10,10,15,5\}$.

KO_{jk}, $j \varepsilon S$, $k \varepsilon U$, is the number of bytes in an owner record occurrence of set j, for set j implemented as k. For WAND $KO_j=\{5,5,10,10,5E_j\}$.

The O_{ij}, M_{ij} and D_i are derived from the conceptual schema. The derivation of A_{ij}, A_i' and A_i'' was discussed above. B, K_0, K_1, C_0, C_1, T are characteristics of the computer/operating system, and the Q_k, KM_k, KO_{jk} are characteristics of the DBMS.

4.4.4 Constraints -

(1) $\sum_{j \varepsilon L} x_{ij} = 1, \; i \varepsilon R$

(Only one location mode per record type.)

(2) $x_{ij} \leq M_{ij}$, $i \varepsilon R, \; j \varepsilon S$

(Can only be VIA set in which record type i is a member.)

(3) $\sum_{k \varepsilon U} y_{jk} = 1, \; j \varepsilon S$

(Only one implementation for a set.)

(4) $\sum_{i \varepsilon R} F_i Z_i \leq pB \leq G_1$

(Database size constraint.)

(5) $bB \leq G_0$

(Core constraint.)

An additional set of constraints could be incorporated to represent upper bounds in elapsed time (response time) for each query used in the design process. Constraint (4) is supported by the following definition of Z_i, which is the total space in bytes required for one occurrence of record type i. The total space requirement is the sum of the space required for the data, overhead, owner pointers, member pointers, CALC pointers, and singular set pointers.

$Z_i = D_i + 15$	data and o'head.
$+ \sum_{j \varepsilon S} O_{ij} (\sum_{k \varepsilon U} KO_k y_{jk})$	owner ptrs
$+ \sum_{j \varepsilon S} M_{ij} (\sum_{k \varepsilon U} KM_k y_{jk})$	member ptrs
$+ 5x_{i,m+1}$	CALC ptrs
$+ 5z_i$	singular set ptrs

We are now ready to define $\bar{A}$, the total number of database accesses independent of buffering strategy, which will then permit us to define the objective function. $\bar{A}$ is derived from accesses to records as members, accesses to records as owners, CALC accesses and cold accesses, either through singular sets or as exhaustive area searches.

4.4.5 Definition Of $\overline{A}$ -

RECALL: $$E_j = \frac{\sum_{i \varepsilon R} M_{ij} F_i}{\sum_{i \varepsilon R} O_{ij} F_i}$$

DEFINE: $$E(p) = \frac{\sum_{i \varepsilon R} F_i Z_i}{pB}$$

E(p) is the % full of the database.

DEFINE: $$PAM_j = \sum_{i \varepsilon R} M_{ij}\left(x_{ij}\left(\left|\frac{z_i E_j}{B(1- E(p))}\right| - 1\right) + (1 - x_{ij})\right)$$

PAM_j is the expected number of <u>physical</u> accesses to visit all members of an occurrence of set type j.

$$\begin{aligned}\overline{A} = \sum_{i \varepsilon R} \Big(& \sum_{j \varepsilon S} M_{ij} A_{ij} PAM_j \\ & + O_{ij} A_{ij} \sum_{k \varepsilon U} y_{jk} (Q_k + (1 - Q_k)(1 + .5\ PAM_j)) \\ & + A_i' z_i F_i + A_i'(1 - z_i)p \\ & + A_i''(1 + f(E(p))) \Big)\end{aligned}$$

4.4.6 Cost Function -

The cost function minimizes total cost (charges) associated with using the database. f_1 Is a buffering function, indicating the probability that the desired record is not already in working storage. One approximation for f_1 is

$$f_1 = 1-(b/p) \quad .$$

f_2 Is a function that returns the average core space occupied. An approximation is

$$f_2 = \text{constant} + bB \quad .$$

The cost function is

$$MIN(\overline{A} \quad f_1 \ T \ f_2 \ C_0 \ + \ p \ C_1) \quad .$$

5.0 ACKNOWLEDGMENT

This research was supported in part by the Office of Naval Research under contract number N00014-75-C-0462.

6.0 BIBLIOGRAPHY

1. "ANSI/X3/SPARC Study Group on Data Base Management Systems Interim Report, 75-02-08," FDT - Bulletin of the ACM SIGMOD, 7, 2, June 1976, pp 97-137.

2. CODASYL, CODASYL Data Base Task Group April 71 Report, available from ACM, New York City.

3. Codd, E. F., "Further Normalization of the Database Relational Model," in Rustin R. (Ed), Database Systems, Prentice Hall, 1972.

4. De, Prabuddha, William D. Haseman and Charles H. Kriebel, "Toward an Optimal Design of a Network Database from Relation Descriptions," Working paper September 1976, Carnegie-Mellon University, Pittsburgh, PA 15213.

5. Gerritsen, Rob, "A Preliminary System for the Design of DBTG Data Structures," CACM 18, 10, October 1975, pp551-557.

6. Gerritsen, Rob, "A Structured DBTG Tutorial," Proceedings, South-Eastern ACM regional conference, Raleigh, North Carolina, April 1975. Also Decision Sciences Working Paper 75-03-01.

7. Gerritsen, Rob, Jim Ribeiro, Ricardo Cortes and Ruth J. Zowader, "WAND User's Guide," Decision Sciences Working Paper 76-01-03, Wharton School, University of Pennsylvania, 1976.

8. Hammer, Michael and Arvola Chan, "Index Selection in a Self-adaptive Data Base Management System," Proceedings 1976 SIGMOD International Conference on Management of Data, ACM, 1976, pp1-8.

9. Martin, James, Computer Data-Base Organization, Prentice-Hall, NJ, 1975.

10. Mitoma, Michael F. and Keki B. Irani, "Automatic Data Base Schema Design and Optimization," Proceedings of the International Conference on Very Large Data Bases, ACM, 1975.

11. Schkolnick, "The Optimal Selection of Secondary Indices for Files," Information Systems, 1, 4, 1975, pp141-146.

12. Severance, D. G., Some Generalized Modeling Structures for Use in Design of File Organizations, Ph.D. Dissertation, University of Michigan, 1974.

13. Tversky, Amos and Daniel Kahneman, "Judgment under Uncertainty: Heuristics and Biases," Science, 185, 27 September, 1974, pp 1124-1131.

14. Yao, S. B. and A. G. Merten, "Selection of File Organization Using an Analytic Model," Proceedings of the International Conference on Very Large Data Bases, ACM, 1975.

DISTRIBUTION LIST

Department of the Navy - Office of Naval Research

Data Base Management Systems Project

Defense Documentation Center (12)
Cameron Station
Alexandria, VA 22314

Office of Naval Research (6)
Arlington, VA 22217

Office of Naval Research
Branch Office, Chicago
536 South Clark Street
Chicago, Illinois 60605

New York Area Office
715 Broadway - 5th Floor
New York, NY 10003

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
(Code RD-1)
Washington, DC 20380

Office of Naval Research
Code 458
Arlington, VA 22217

Mr. E. H. Gleissner
Naval Ship Research and
Development Center
Computation & Mathematics Dept.
Bethesda, MD 20084

Mr. Kim B. Thompson
Technical Director
Information Systems Division
(OP-911G)
Office of Chief of Naval Operations
Washington, DC 20350

Professor Omar Wing
Columbia University
Dept of Electrical Engineering
and Computer Science
New York, NY 10027

Office of Naval Research (2)
Information Systems Program
Code 437
Arlington, VA 22217

Office of Naval Research
Code 102IP Branch Office, Boston
495 Summer Street
Boston, MA 02210

Office of Naval Research
Branch Office, Pasadena
1030 East Green Street
Pasadena, CA 91106

Naval Research Laboratory (6)
Technical Information Division
Code 2627
Washington, DC 20375

Office of Naval Research
Code 455
Arlington, VA 22217

Naval Electronics Laboratory Center
Advanced Software Technology Division
Code 5200
San Diego, CA 92152

Captain Grace M. Hopper
NAICOM/MIS Planning Branch
(OP-916D)
Office of Chief of Naval Operations
Washington, DC 20350

Bureau of Library and
Information Science Research
Rutgers - The State University
189 College Avenue
New Brunswick, NJ 08903
Attn: Dr. Henry Voos